实验4  
实现支持向量机

指导教师：刘胥影 助教：田秋雨

2024年5月23日

Due：2023年6月6日

## 任务描述

通过两种方式实现SVM：

1. 手动实现SMO算法，并与直接使用传统二次规划方法进行对比，
2. 通过scikit-learn库实现软间隔SVM。

并在breast cancer数据集上进行验证与实验。该数据集是一个二分类问题，属性均为连续属性，并已进行标准化。

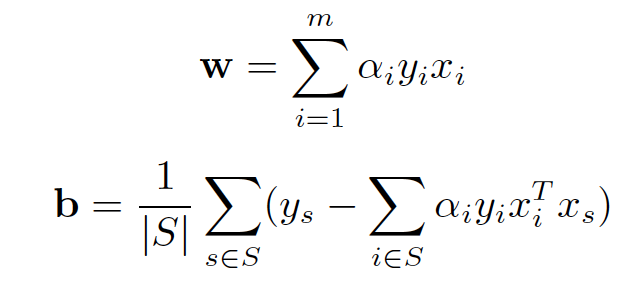
## 教学要求

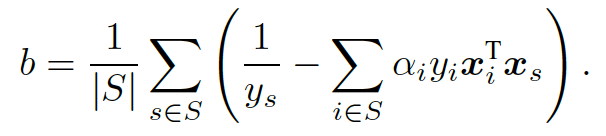
1. 掌握硬间隔SVM和软间隔SVM及其核化方法的原理
2. 掌握SMO算法
3. 了解基于Python语言的scikit-learn（sklearn）库，掌握本实验涉及的相关部分
4. 进行参数选择以及参数分析实验，理解正则化常数C以及核函数的影响

## 实验要求

**1. 手动实现SVM的SMO算法**

SVM的对偶问题实际是一个二次规划问题，除了SMO算法外，传统二次规划方法也可以用于求解对偶问题。求得最优拉格朗日乘子后，超平面参数***w***, *b*可由以下式子得到：





请完成以下任务：

1. [10pts] 不考虑软间隔情况，直接使用传统二次规划（QP）方法求解（实现）训练集上的硬间隔SVM对偶问题。观察并回答，这样的求解方法是否会出现问题，为什么？
2. [40pts] 不限定硬间隔或是软间隔SVM，也不限定是否为核化SVM，根据需要选择合适的方法，手动实现SMO算法[[1]](#footnote-1)求解SVM对偶问题。注意第3步，KKT条件验证步骤不能缺少。
3. [10pts] 对测试数据进行预测，确保预测结果尽可能准确。

**2. 使用sklearn库简洁实现软间隔SVM**

1. [20pts] 使用sklearn库简洁实现软间隔SVM。首先实现以下4个示例性的SVM模型：

* 线性SVM：正则化常数C=1，核函数为线性核，
* 线性SVM：正则化常数C=1000，核函数为线性核，
* 非线性SVM：正则化常数C=1，核函数为多项式核，d=2，
* 非线性SVM：正则化常数C=1000，核函数为多项式核，d=2，

观察并比较它们在测试集上的性能表现。

1. [20pts] 参数选择与参数分析

参数的选择对SVM的性能有很大的影响。确定正则化常数C与核函数及其参数的选择范围（可以在以下常用核函数中选择一种或多种），选用合适的实验评估方法（回顾第2章的内容，如K折交叉验证法等）进行参数选择，并进行参数分析实验。

* 正则化常数C的选择范围可以是以下数量级，如：
* 核函数
  + 线性核：，无参数，退化为SVM基本型
  + 多项式核：,
    - *c*: 偏移量，通常取值0或1
    - *d*: 多项式的次数，通常取值2, 3, 4等较小的整数，取值为1时退化为线性核
  + 多项式核另外一种常用的形式：
    - : 用于控制核函数的影响范围，常用值为1/n\_features, n\_features是属性个数，或者也可以尝试0.001, 0.01, 0.1, 1, 10等
    - *c*: 偏移量，通常取值0或1
    - *d*: 多项式的次数，通常取值2, 3, 4等较小的整数，取值为1时退化为线性核
  + 高斯核/RBF核：
    - ：带宽参数，控制核函数的平滑程度。常用取值范围为到。典型值为1/(n\_features\* X.var())，其中X.var()表示数据集X的方差，这可以确保 𝛾 的值在数据特征的数量和数据的方差范围内适当缩放。
  + Laplace核：
    - ：带宽参数，控制核函数的平滑程度。常用取值范围参考高斯核的 值
  + Sigmoid核函数：
    - ：缩放因子，常用取值范围为到
    - c: 偏移量，通常取值0或1

请注意，SVM中的参数通常需要联合选择，特别是对于核函数的参数与正则化参数C。参数联合优化的策略有：

1. 网格搜索：网格搜索是一种穷举搜索法，在给定的参数空间中逐一尝试每种可能的参数组合，并选用合适的评估方法进行评估。
2. 随机搜索：在参数空间中对参数组合的随机采样进行评估。通常可以在较大的参数空间中找到接近最优的参数组合。步骤如下：

* 定义参数的取值范围
* 随机采样若干参数组合
* 对每个采样的参数组合进行评估
* 选择性能最优的参数组合

1. 贝叶斯优化：贝叶斯优化是一种更为高级的优化方法，通过构建一个模型来估计参数空间的性能分布，并通过最大化预期改进（Expected Improvement, EI）等准则来选择下一组参数进行评估。其步骤如下：

* 初始采样若干参数组合并进行评估
* 构建模型（如高斯过程回归）
* 基于模型选择下一组参数进行评估
* 更新模型并重复第3个步骤，直到达到预定的评估次数或满足停止条件。

本实验中可以选择较为简单的网格搜索策略或随机搜索策略进行调参。

参数分析实验需要报告：

* 评估方法
* 参数取值范围
* 搜索策略
* 比较分析每组参数上的性能情况
* 最终选择的最优参数

## 提交代码与报告

其中报告内容包括以下几个部分：

1. 问题复述
2. 代码思路
3. 问题解答（针对原问题的解答，有问题需要回答问题，没有问题需要用一两句话总结）
4. 代码附录（放在最后）

## 提交要求

1.**.ipynb的jupyter notebook，或者.py的脚本文件。**

2. 文件载入使用**相对路径**。**！！！！！！！**

示例：X\_train=np.loadtxt("breast\_cancer\_Xtrain.csv",delimiter=',')

3. 一二两大问分文件作答。

## 代码样例

import numpy as np

from cvxopt import matrix, solvers

def svm\_qp(X, y):

m, n = X.shape

y = y.astype(float)

# Create the P, q, G, h matrices for the QP solver

K = np.dot(X, X.T) \* np.outer(y, y)

P = matrix(K)

q = matrix(-np.ones((m, 1)))

G = matrix(np.diag(-np.ones(m)))

h = matrix(np.zeros(m))

A = matrix(y.reshape(1, -1))

b = matrix(np.zeros(1))

# Solve QP problem

sol = solvers.qp(P, q, G, h, A, b)

alphas = np.array(sol['x']).flatten()

# Compute weight vector w

w = np.sum(alphas \* y[:, None] \* X, axis=0)

# Compute bias term b

support\_vectors = (alphas > 1e-5)

b = np.mean(y[support\_vectors] - np.dot(X[support\_vectors], w))

return w, b, alphas

# Example usage

X = np.array([[2, 3], [3, 3], [4, 3], [1, 1], [2, 1], [3, 1]])

y = np.array([1, 1, 1, -1, -1, -1])

w, b, alphas = svm\_qp(X, y)

print("Weights:", w)

print("Bias:", b)

class SVM:

def \_\_init\_\_(self, C=1.0, tol=1e-3, max\_passes=5):

self.C = C

self.tol = tol

self.max\_passes = max\_passes

def fit(self, X, y):

m, n = X.shape

self.X = X

self.y = y

self.alphas = np.zeros(m)

self.b = 0

self.E = np.zeros(m)

self.K = np.dot(X, X.T)

passes = 0

while passes < self.max\_passes:

num\_changed\_alphas = 0

for i in range(m):

self.E[i] = self.\_E(i)

if (y[i] \* self.E[i] < -self.tol and self.alphas[i] < self.C) or \

(y[i] \* self.E[i] > self.tol and self.alphas[i] > 0):

j = np.random.choice([\_ for \_ in range(m) if \_ != i])

self.E[j] = self.\_E(j)

alpha\_i\_old = self.alphas[i].copy()

alpha\_j\_old = self.alphas[j].copy()

if y[i] != y[j]:

L = max(0, self.alphas[j] - self.alphas[i])

H = min(self.C, self.C + self.alphas[j] - self.alphas[i])

else:

L = max(0, self.alphas[j] + self.alphas[i] - self.C)

H = min(self.C, self.alphas[j] + self.alphas[i])

if L == H:

continue

eta = 2 \* self.K[i, j] - self.K[i, i] - self.K[j, j]

if eta >= 0:

continue

self.alphas[j] -= y[j] \* (self.E[i] - self.E[j]) / eta

self.alphas[j] = np.clip(self.alphas[j], L, H)

if abs(self.alphas[j] - alpha\_j\_old) < 1e-5:

continue

self.alphas[i] += y[i] \* y[j] \* (alpha\_j\_old - self.alphas[j])

b1 = self.b - self.E[i] - y[i] \* (self.alphas[i] - alpha\_i\_old) \* self.K[i, i] - \

y[j] \* (self.alphas[j] - alpha\_j\_old) \* self.K[i, j]

b2 = self.b - self.E[j] - y[i] \* (self.alphas[i] - alpha\_i\_old) \* self.K[i, j] - \

y[j] \* (self.alphas[j] - alpha\_j\_old) \* self.K[j, j]

if 0 < self.alphas[i] < self.C:

self.b = b1

elif 0 < self.alphas[j] < self.C:

self.b = b2

else:

self.b = (b1 + b2) / 2

num\_changed\_alphas += 1

if num\_changed\_alphas == 0:

passes += 1

else:

passes = 0

self.w = np.dot((self.alphas \* y).T, X)

def \_E(self, i):

return self.b + np.dot(self.alphas \* self.y, self.K[:, i]) - self.y[i]

def predict(self, X):

return np.sign(np.dot(X, self.w) + self.b)

# Example usage

X = np.array([[2, 3], [3, 3], [4, 3], [1, 1], [2, 1], [3, 1]])

y = np.array([1, 1, 1, -1, -1, -1])

svm = SVM(C=1.0)

svm.fit(X, y)

predictions = svm.predict(X)

print("Predictions:", predictions)

# Assuming X\_test is your test data

X\_test = np.array([[2, 2], [3, 2], [4, 2], [1, 0], [2, 0], [3, 0]])

predictions = svm.predict(X\_test)

print("Test Predictions:", predictions)

1. SMO算法的详细步骤可以参考”王贝伦《机器学习》“4.7.2.2节P169。参见实验文档中的相关pdf文件。 [↑](#footnote-ref-1)